

AD-A034 266      SYRACUSE UNIV N Y DEPT OF INDUSTRIAL ENGINEERING AND--ETC F/G 9/2  
TIMING FIGURES FOR INVERTING LARGE MATRICES CONTAINING COMPLEX --ETC(U)  
NOV 76 P B BERRA, A K SINGHANIA      F30602-75-C-0121

UNCLASSIFIED

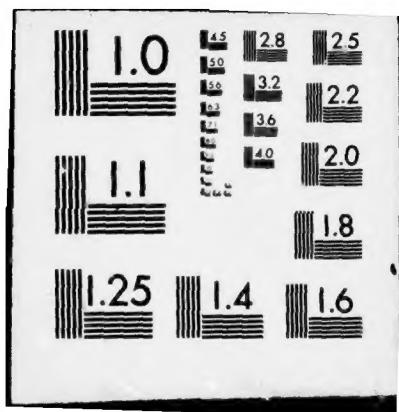
RADC-TR-76-339

NL

1 OF 1  
ADA  
034 266



END  
DATE  
FILMED  
2-16-77  
NTIS



**U.S. DEPARTMENT OF COMMERCE  
National Technical Information Service**

**AD-A034 266**

**TIMING FIGURES FOR INVERTING LARGE MATRICES  
CONTAINING COMPLEX NUMBERS USING THE STARAN  
ASSOCIATIVE PROCESSOR**

**SYRACUSE UNIVERSITY, NEW YORK**

**NOVEMBER 1976**

ADA 034266

014063

RADC-TR-76-339  
Technical Report  
November 1976



TIMING FIGURES FOR INVERTING LARGE MATRICES  
CONTAINING COMPLEX NUMBERS  
USING THE STARAN ASSOCIATIVE PROCESSOR

Syracuse University

Approved for public release;  
distribution unlimited.

D D C  
REPRODUCED  
JAN 14 1977  
A

ROME AIR DEVELOPMENT CENTER  
AIR FORCE SYSTEMS COMMAND  
GRIFFISS AIR FORCE BASE, NEW YORK 13441

REPRODUCED BY  
NATIONAL TECHNICAL  
INFORMATION SERVICE  
U. S. DEPARTMENT OF COMMERCE  
SPRINGFIELD, VA. 22161

This report has been reviewed by the RADC Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public including foreign nations.

This report has been reviewed and is approved for publication.

APPROVED:

*Kenneth R. Starkiewicz*

KENNETH R. STARKIEWICZ  
Project Engineer

APPROVED:

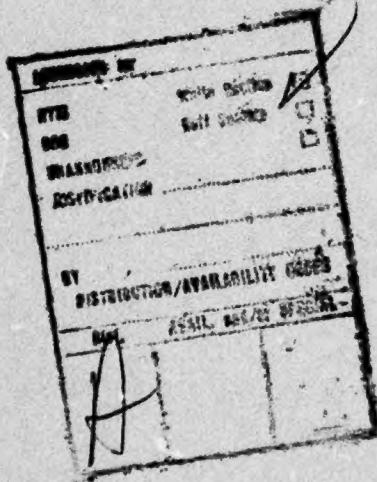
*Joseph J. Naresky*

JOSEPH J. NARESKY  
Chief, Reliability & Compatibility Division

FOR THE COMMANDER:

*John P. Huss*

JOHN P. HUSS  
Acting Chief, Plans Office



Do not return this copy. Retain or destroy.

i (a)

## UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-76-339	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) TIMING FIGURES FOR INVERTING LARGE MATRICES CONTAINING COMPLEX NUMBERS USING THE STARAN ASSOCIATIVE PROCESSOR		5. TYPE OF REPORT & PERIOD COVERED Technical Report June 1975 - June 1976
7. AUTHOR(s) P. Bruce Berra Ashok K. Singhania		6. PERFORMING ORG. REPORT NUMBER N/A
9. PERFORMING ORGANIZATION NAME AND ADDRESS Syracuse University/Dept of Industrial Engineering and Operations Research Syracuse NY 13210		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62702F 45400132
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (RBCT) Griffiss AFB NY 13441		12. REPORT DATE November 1976
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same		13. NUMBER OF PAGES 70
		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A
16. DISTRIBUTION STATEMENT (of this Report)  Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)  Same		
18. SUPPLEMENTARY NOTES RADC Project Engineer: Kenneth R. Siarkiewicz (RBCT)		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Associative Processor Matrix Inversion Complex Numbers Timing Equations Electromagnetic Compatibility		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This report is a follow on to RADC-TR-75-73 "Timing Figures for Inverting Large Matrices Using the STARAN Associative Processor." In that report timing equations were developed for matrices consisting of real numbers whereas this report deals with complex numbers. Algorithms and timing equations are developed for matrices of rank up to 511 and matrices with $512 < \text{rank} < 1023$ . Results indicate that it takes 2 1/2 to 4 times longer to invert matrices consisting of complex numbers than it does for real numbers.		

TABLE OF CONTENTS

	Page
<b>ABSTRACT.....</b>	<b>i</b>
<b>LIST OF FIGURES.....</b>	<b>iii</b>
<b>LIST OF TABLES.....</b>	<b>iv</b>
<b>Introduction.....</b>	<b>1</b>
<b>Rectangular Coordinates.....</b>	<b>2</b>
<b>Polar Coordinates.....</b>	<b>6</b>
<b>Coordinate Conversions.....</b>	<b>11</b>
<b>Inversion of Complex Matrices Using an Associative             Processor (AP).....</b>	<b>15</b>
<b>Case A.....</b>	<b>22</b>
<b>Algorithm 1.....</b>	<b>23</b>
<b>Case B.....</b>	<b>39</b>
<b>Algorithm 2.....</b>	<b>39</b>
<b>Development of Timing Expressions.....</b>	<b>49</b>
<b>CONCLUSIONS.....</b>	<b>59</b>
<b>REFERENCES.....</b>	<b>63</b>

## LIST OF FIGURES

	Page
<b>FIGURE 1:</b> Inversion of a $3 \times 3$ Matrix With Complex Elements.....	16
<b>FIGURE 2:</b> AP Memory Maps, Case A.....	24
<b>FIGURE 3:</b> $\vec{x}_n$ and $\vec{y}_n$ Vectors Corresponding To Matrix A of Figure 1.....	33
<b>FIGURE 4.1:</b> Initial AP Memory Map During Inversion of the A Matrix Using Algorithm 1.....	34
<b>FIGURE 4.2:</b> Intermediate AP Memory Map.....	35
<b>FIGURE 4.3:</b> Intermediate AP Memory Map.....	36
<b>FIGURE 4.4:</b> Intermediate AP Memory Map.....	37
<b>FIGURE 5:</b> AP Memory Maps, Case B.....	40
<b>FIGURE 6:</b> Matrix Inversion Time with 4 Arrays and Rank < 511.....	53
<b>FIGURE 7:</b> Matrix Inversion Time with 4 Arrays and Rank < 1023.....	54
<b>FIGURE 8:</b> Matrix Inversion Time with 16 Arrays and Rank < 2000.....	57

LIST OF TABLES

	Page
TABLE 1: Execution Times ( $\mu$ secs) vs. Coordinate Representation. (STARAN time values used).....	14
TABLE 2: Memory and Arithmetic Operation Times for STARAN.....	50
TABLE 3: Timing Data for Inverting a Matrix with Complex Numbers for $25 < \text{Rank} < 511$ . (4 Arrays)..	55
TABLE 4: Timing Data for Inverting a Matrix with Complex Numbers for $25 < \text{Rank} < 1023$ . (4 Arrays). .	56
TABLE 5: Timing Data for Inverting a Matrix with Complex Numbers for $100 < \text{Rank} < 2000$ . (16 Arrays).....	58
TABLE 6: Comparative Inversion Times for Matrices with Complex Numbers and Matrices with Real Numbers..	60

### Introduction

This report is a follow on to RADC-TR-75-73 "Timing Figures for Inverting Large Matrices Using the STARAN Associative Processor" [1]. In that report timing figures were developed for the inversion of the A matrix in the system of linear equations

$$AX = B$$

for matrices with rank up to 2000. The assumption was made that the A matrix contained only real numbers. In this report we follow the same methodology but assume that the A matrix is composed of complex numbers. For suitable introductory material on associative processors, associative processor applications and matrix inversion refer to the previous report which is available from the Defense Documentation Center in Alexandria, Virginia. (AD A009643).

In this report we first look at performing division, multiplication and subtraction in rectangular coordinates, then polar coordinates and finally a combination of the two. We then choose rectangular coordinates and develop timing equations for the case in which the rank of the

matrix can be as high as 511 and the case where the rank is between 512 and 1023. Finally, we utilize STARAN timing data to develop plots of total time to invert the matrix versus the rank of the matrix.

#### Rectangular Coordinates

Recall from reference 1 that inverting a matrix of rank N requires N divisions,  $N(N-1)$  multiplications and  $N(N-1)$  subtractions.

Suppose we have a set of complex numbers  $x_j + iy_j$  for  $j = 1, 2, \dots, N$  and would like to divide these numbers in parallel by one complex number from within the set. Call the divisor  $x + iy$ . Then, in order to perform the division, we must form the complex conjugate and multiply as follows:

$$\frac{(x_j + iy_j)(x - iy)}{(x + iy)(x - iy)} = \frac{x_j x + y_j y}{x^2 + y^2} + i \frac{xy_j - x_j y}{x^2 + y^2}$$

If we assume that the  $x_j$  are in one column of the array and the  $y_j$  are in another column then it will require four comparand to field multiplies to form the following products:

$x_j x$ ,  $xy_j$ ,  $y_j y$  and  $x_j y$ .

We can then perform one addition to form:

$$x_j x + y_j y$$

and one subtraction to form:

$$xy_j - x_j y.$$

We then must divide each of the above by:

$$x^2 + y^2.$$

Note that this sum is already available since  $x$  and  $y$  are contained in the set of numbers. We thus obtain the time for division of one column of complex numbers by a single complex number to be:

$$T_{DR} = 4 t_{mc} + t_a + t_s + 2 t_{dc}$$

where  $t_{mc}$  is the comparand to field multiply time,  $t_a$  is the field to field add time,  $t_s$  is the field to field subtract time and  $t_{dc}$  is the comparand to field divide time. If we use STARAN time values we obtain:

$$T_{DR} = 4(715) + 412 + 412 + 2(772) = 5228 \mu\text{sec.}$$

In all of the calculations using STARAN timing values we will assume that the floating point arithmetic capability is implemented in software and that we are operating with 32 bit field lengths. Furthermore, it should be pointed out that the time to perform data movement and other non-arithmetic operations has been omitted.

Using similar assumptions consider the multiplication of a column of complex numbers by a complex number:

$$(x_j + iy_j)(x + iy) = (x_jx - y_jy) + i(x_jy + xy_j).$$

In this case we have four comparand to field multiplies to form the cross products, one addition and one subtraction. We thus obtain:

$$T_{MR} = 4 t_{mc} + t_a + t_s.$$

For STARAN time values we obtain

$$T_{MR} = 4(715) + 412 + 412 = 3684 \mu\text{sec.}$$

Subtraction is by far the least time consuming. For a field to field subtraction we have:

$$(x_j + iy_j) - (x_k + iy_k) = (x_j - x_k) + i(y_j - y_k)$$

which results in only two subtractions or

$$T_{SR} = 2 t_s.$$

For STARAN time values we obtain:

$$T_{SR} = 2(412) = 824 \mu\text{sec.}$$

If we now consider N divisions, N(N-1) multiplications and N(N-1) subtractions we obtain the total arithmetic time as:

$$T_R = NT_{DR} + N(N-1) T_{MR} + N(N-1) T_{SR}.$$

For STARAN timing values this reduces to:

$$\begin{aligned} T_R &= N(5228 + (N-1)(3684 + 824)) \\ &= 4508N^2 + 720N \mu\text{sec.} \end{aligned}$$

As an example suppose the rank of the matrix is 500, then  
 $T_R \approx 1127$  seconds.

#### Polar Coordinates

Now suppose the complex numbers are in polar coordinates such that we have numbers  $r_j e^{i\theta_j}$  for  $j = 1, 2, \dots, N$  and would like to divide and multiply these numbers by another complex number  $re^{i\theta}$  which is again a member of the original set. Also, we would like to subtract two columns of complex numbers.

Considering division we have:

$$\frac{r_1 e^{i\theta_1}}{r e^{i\theta}} = \frac{r_1}{r} e^{i(\theta_1 - \theta)}$$

Again, assuming that the  $r_j$  and  $\theta_j$  are in two columns of the array the time it would take to perform a division of complex numbers is determined by one comparand to field divide and one comparand from field subtraction. Thus, we obtain

$$T_{DP} = t_{dc} + t_{sc}$$

or for STARAN  $T_{DP} = 772 + 412 = 1184 \mu\text{sec.}$

For multiplication we must obtain

$$(r_j e^{i\theta_j}) (r e^{i\theta}) = r_j r e^{i(\theta_j + \theta)}$$

or one comparand to field multiply and one comparand to field addition.

We thus obtain

$$T_{MP} = t_{mc} + t_{ac}$$

or for STARAN  $T_{MP} = 715 + 412 = 1127 \mu\text{sec.}$

While there appears to be considerable time advantage in using polar coordinates for division and multiplication the advantage disappears when subtraction is considered. Consider the difference between the following columns of numbers:

$$r_j e^{i\theta_j} - r_k e^{i\theta_k}$$

In order to take this difference we must first transform these numbers to their equivalent sine and cosine representations. Thus, we obtain the following:

$$r_j e^{i\theta_j} = r_j (\cos \theta_j + i \sin \theta_j)$$

$$r_k e^{i\theta_k} = r_k (\cos \theta_k + i \sin \theta_k)$$

and

$$r_j e^{i\theta_j} - r_k e^{i\theta_k} = (r_j \cos \theta_j - r_k \cos \theta_k)$$

$$+ i(r_j \sin \theta_j - r_k \sin \theta_k)$$

In order to perform these calculations we must perform two cosine functions, two sine functions, four multiplies and two subtractions. Now, given these results we must transfer back to our previous polar coordinate form. For ease of notation suppose we let X and Y be the following:

$$X + iY = (r_j \cos \theta_j - r_k \cos \theta_k)$$

$$+ i(r_j \sin \theta_j - r_k \sin \theta_k).$$

The result that we are seeking is the following in polar coordinates:

$$\left( \sqrt{x^2 + y^2}, e^{i \tan^{-1}(\frac{y}{x})} \right)$$

In order to perform this operation we must reproduce X and Y, multiply them together to get  $X^2$  and  $Y^2$ , take the square root, divide Y by X and take an arctan. When we put all of these operations together we have the following equation for the subtraction of two columns in polar coordinates:

$$T_{SP} = 2 t_{cos} + 2 t_{sin} + 6 t_m + 2 t_s + t_d + t_a + t_{arctan} + t_{sqrt}$$

where  $t_{cos}$  is the time to take the cosine of a column of numbers in the array;  $t_{sin}$  is the time to take the sine;  $t_a$ ,  $t_s$ ,  $t_m$  and  $t_d$  are the field to field add, subtract, multiply and divide times respectively;  $t_{arctan}$  is the time required to take the arctangent of a column of numbers; and  $t_{sqrt}$  is the square root time. The time to reproduce the X and Y columns has been omitted since it is small in comparison to the other times.

If we utilize STARAN timing values we obtain the following time

$$T_{SP} = 2(12363) + 2(12363) + 6(832) + 2(412)$$

$$+ 890 + 412 + 12363 + 652 = 69,585 \mu\text{sec.}$$

Now, if we add together the division, multiplication and subtraction times we obtain:

$$T_p = NT_{DP} + N(N-1) T_{MP} + N(N-1) T_{SP}.$$

For STARAN timing values this reduces to

$$T_p = N(1184 + (N-1)(1127 + 69,585))$$

$$= 70712 N^2 - 69528 N \mu\text{sec}$$

If we again consider a matrix of rank = 500 we obtain  $T_p = 17643$  sec.

It is apparent that the large time required for the inversion process when polar coordinates are used is mainly due to the underlying subtraction processes that are involved. The subtraction process requires the use of the sine and cosine functions and these in turn require enough execution time to more than offset the savings that are made during the division and multiplication processes.

### Coordinate Conversions

To complete our discussion of which complex number representation scheme is most suitable for matrix inversion purposes, let us now consider the two other cases that are possible, viz: start with the matrix elements represented in rectangular coordinates and go ahead with the inversion process, converting to polar representation and back when deemed desirable, and vice versa. We now consider how these schemes perform in terms of inversion times.

From the discussion of the subtraction process involved with polar coordinates, it is clear that conversion of a set of numbers in rectangular coordinates to polar coordinates, i.e.,

$$(x, y) \longrightarrow (\sqrt{x^2 + y^2}, \tan^{-1} \frac{y}{x})$$

requires an amount of time equal to:

$$T_{RP} = 2 t_m + t_d + t_a + t_{\arctan} + t_{\sqrt{}}.$$

For STARAN timing values this reduces to:

$$\begin{aligned} T_{RP} &= 2(832) + 890 + 412 + 12363 + 652 \\ &= 15981 \mu \text{secs.} \end{aligned}$$

Also, conversion of a set of numbers from polar coordinates to rectangular coordinates, i.e.,

$$(r, \theta) \longrightarrow (r \cos \theta, r \sin \theta)$$

would require an amount of time equal to:

$$T_{PR} = t_{\cos} + t_{\sin} + 2 t_m.$$

For STARAN timing values this reduces to:

$$T_{PR} = 2(12363) + 2(832)$$

$$= 26390 \mu\text{secs.}$$

If we represent the results obtained above in tabular form, as shown in Table 1, we find that working with rectangular coordinates clearly outperforms the case where polar coordinates are used. Although the use of polar coordinates results in some savings in time for division and multiplication operations, these savings are more than offset during the subtraction process, because of the large amounts of time required

in invoking the sine and cosine functions which are needed when polar coordinates are used. For the same reason, starting with polar coordinates and changing to rectangular coordinates as and when needed is not attractive at all. Similarly, starting with rectangular coordinates and converting to polar coordinates for division and multiplication, offers no savings because of the large times involved with any attempts at coordinate conversion. It is thus evident that the best course of action is to perform all required operations in rectangular coordinates and, therefore, in all that follows we will assume that the matrix elements are represented in rectangular coordinates and stored as such in mass storage.

TABLE 1: Execution Times ( $\mu$ sec) vs. Coordinate

Representation. (STARAN time values used)

	Rectangular Coordinates	Polar Coordinates
Division	5228	1184
Multiplication	3684	1127
Subtraction	824	69585
Matrix Inversion	$4508 N^2 + 720 N$	$70712 N^2 - 69528 N$
Conversion to	26390	15981

### Inversion of Complex Matrices Using an Associative Processor (AP).

The process of inverting matrices with real elements on an AP was discussed in [1]. In principle, we can use the same Gauss-Jordan method for inverting matrices with complex elements, except that the algorithms have to be modified to reflect the fact that the arithmetic operations have complex numbers as their arguments, etc. As discussed in [1], for reasons of storage efficiency, we want to start off with only the first row of the identity matrix appended to the top of the given matrix, instead of appending the entire identity matrix to the given matrix. We have already made the assumption that the matrix elements are available to us in rectangular coordinates; we now make the additional assumption that the first row of the identity matrix has been appended to the top of the given matrix in mass storage, so that we shall consider the matrix ( $Z$ ), as it resides in mass storage, to be  $(N + 1)$  rows by  $N$  columns.

Shown in Figure 1 is the solution of a  $3 \times 3$  matrix with complex numbers. The technique that is utilized to determine the inverse of this matrix on an AP will be illustrated in some detail subsequently.

$$\text{Let } A = \begin{bmatrix} 1 + i1 & -2 - i2 & 3 + i3 \\ 0 & 2 + i2 & -3 - i3 \\ 1 + i1 & 1 + i1 & 1 + i1 \end{bmatrix}$$

We would then start initially with:

$$Z = \begin{bmatrix} 1 + i0 & 0 + i0 & 0 + i0 \\ \dots & \dots & \dots \\ 1 + i1 & -2 - i2 & 3 + i3 \\ 0 + i0 & 2 + i2 & -3 - i3 \\ 1 + i1 & 1 + i1 & 1 + i1 \end{bmatrix} \text{First row of identity matrix.}$$

On successive iterations of the inversion process we obtain:

$$\begin{bmatrix} 0.5 - i0.5 & 0 + i0 & 0 + i0 \\ 1 + i0 & -2 - i2 & 3 + i3 \\ 0 + i0 & 2 + i2 & -3 - i3 \\ 1 + i0 & 1 + i2 & 1 + i1 \end{bmatrix} \text{After dividing column 1 (the pivot) by } 1 + i1.$$

$$\begin{bmatrix} 0.5 - i0.5 & 2 + i0 & -3 + i0 \\ 1 + i0 & 0 + i0 & 0 + i0 \\ 0 + i0 & 2 + i2 & -3 - i3 \\ 1 + i0 & 3 + i3 & -2 - i2 \end{bmatrix} \text{After performing elementary column operations to normalize columns 2 and 3 with respect to the pivot.}$$

FIGURE 1: Inversion of a  $3 \times 3$  Matrix With Complex Elements.

$$\left[ \begin{array}{ccc} 0.5 - i0.5 & 2 + i0 & -3 + i0 \\ 0 + i0 & 1 + i0 & 0 + i0 \\ \hline 0 + i0 & 2 + i2 & -3 - i3 \\ 1 + i0 & 3 + i3 & -2 - i2 \end{array} \right] \quad \begin{array}{l} \text{After replacing the second} \\ \text{row with the second row of} \\ \text{the identity matrix} \end{array}$$

$$\left[ \begin{array}{ccc} 0.5 - i0.5 & 0.5 - i0.5 & -3 + i0 \\ 0 + i0 & 0.25 - i0.25 & 0 + i0 \\ \hline 0 + i0 & 1 + i0 & -3 - i3 \\ 1 + i0 & 1.5 + i0 & -2 - i2 \end{array} \right] \quad \begin{array}{l} \text{Second column becomes the} \\ \text{pivot} \end{array}$$

$$\left[ \begin{array}{ccc} 0.5 - i0.5 & 0.5 - i0.5 & 0 + i0 \\ 0 + i0 & 0.25 - i0.25 & 1.5 + i0 \\ \hline 0 + i0 & 1 + i0 & 0 + i0 \\ 1 + i0 & 1.5 + i0 & 2.5 + i2.5 \end{array} \right]$$

$$\left[ \begin{array}{ccc} 0.5 - i0.5 & 0.5 - i0.5 & 0 + i0 \\ 0 + i0 & 0.25 - i0.25 & 1.5 + i0 \\ \hline 0 + i0 & 0 + i0 & 1 + i0 \\ 1 + i0 & 1.5 + i0 & 2.5 + i2.5 \end{array} \right] \quad \begin{array}{l} \text{Third row of identity matrix} \end{array}$$

FIGURE 1; Inversion of a  $3 \times 3$  Matrix with Complex Elements  
(Continued)

$$\left[ \begin{array}{ccc} 0.5 - i0.5 & 0.5 - i0.5 & 0 + i0 \\ 0 + i0 & 0.25 - i0.25 & 0.3 - i0.3 \\ 0 + i0 & 0 + i0 & 0.2 - i0.2 \\ 1 + i0 & 1.5 + i0 & 1 + i0 \end{array} \right] \quad \begin{matrix} \text{Third column becomes} \\ \text{the pivot} \end{matrix}$$

$$\left[ \begin{array}{ccc} 0.5 - i0.5 & 0.5 - i0.5 & 0 + i0 \\ -3 + i0.3 & -0.2 + i0.2 & 0.3 - i0.3 \\ -0.2 + i0.2 & -0.3 + i0.3 & 0.2 - i0.2 \\ \hline 0 + i0 & 0 + i0 & 1 + i0 \end{array} \right]$$

$$\therefore A^{-1} = \left[ \begin{array}{ccc} 0.5 - i0.5 & 0.5 - i0.5 & 0 + i0 \\ -0.3 + i0.3 & -0.2 + i0.2 & 0.3 - i0.3 \\ -0.2 + i0.2 & -0.3 + i0.3 & 0.2 - i0.2 \end{array} \right]$$

FIGURE 1: Inversion of a  $3 \times 3$  Matrix With Complex Elements

(Continued)

For the general case, let the matrix, Z be represented as:

$$\begin{aligned}Z &= [\vec{z}_1 \ \vec{z}_2 \ \dots \ \vec{z}_N] \\&= [\vec{x}_1 + i \vec{y}_1 \ \vec{x}_2 + i \vec{y}_2 \ \dots \ \vec{x}_N + i \vec{y}_N],\end{aligned}$$

where,  $Z$  is the  $(N+1) \times N$  matrix with complex elements,

$\vec{z}_n$  is the  $n$ th column of  $Z$ ,

$\vec{x}_n$  are the real components of  $\vec{z}_n$ ,

$\vec{y}_n$  are the imaginary components of  $\vec{z}_n$ ,

and  $n = 1, \dots, N$ .

We can thus think of the matrix elements as constituting  $2N$  column vectors ( $\vec{x}_n$  and  $\vec{y}_n$ ) of length  $(N+1)$  each. We now make a third assumption, viz. the vectors  $\vec{x}_n$  and  $\vec{y}_n$ , can be individually retrieved from the mass storage when desired for loading onto the associative arrays. This is tantamount to saying that we can address the vectors  $\vec{x}_n$  and  $\vec{y}_n$  individually for moving them to and from the mass storage and the associative arrays, such that we can directly load  $\vec{x}_n$  and  $\vec{y}_n$  into two separate fields of an associative array, or, in the same field, one below the other.

Before beginning the discussion of the inversion algorithm, we must introduce some additional notation. Let

$W$  = Number of words in an associative array

$A$  = Number of associative arrays

$L = \left\lfloor \frac{WA}{2(N+1)} \right\rfloor =$  largest integer in  $\frac{WA}{2(N+1)}$

= Number of  $(\vec{x}_n, \vec{y}_n)$  vector pairs (corresponding to matrix columns) that can be loaded in an AP field,

$M = L \lfloor (N-1) =$  the smaller of  $L$  and  $(N-1)$ ,

$K = \lceil \frac{(N-1)}{M} \rceil =$  smallest integer greater than or equal to  $\frac{(N-1)}{M}$

Proceeding with the inversion process as in [1], we want to bring the first column (real and imaginary) of the matrix into one field of the AP and then process each of the remaining  $(N-1)$  columns of the matrix relative to the first column, according to the Gauss-Jordan Elimination Method. We next use the second column of the partly processed matrix as the pivot and process the other  $(N-1)$  columns relative to it, and so on. In order to process as many columns of the matrix in parallel against a pivot as we can, we must load as many columns of the matrix as possible into the same field of the AP. A maximum of  $L$  matrix columns (for  $N \times N$  matrices with complex elements, and with the first row of the identity matrix appended to their top) can be loaded in an AP field. However, if

(N-1) is less than L, we need to load only those (N-1) columns in the associative array field for processing against the pivot. M is thus the number of matrix columns to be actually loaded into an AP field for processing against the pivot. After these M columns have been processed, we get and process M more columns, if there remain that many, and so on. A total of K iterations would thus be required to process the (N-1) columns relative to the given pivot.

Unless otherwise stated we will work with the assumption that the AP has 1024 words (viz. the RADC STARAN with 4 arrays). It is evident then that as long as the rank of the matrix is less than or equal to 511, (or more generally if  $N < \frac{WA}{2}$ ) one column (or more) of the matrix can entirely fit into a field of the AP. However, if the rank of the matrix is 512 or more, not even one entire column can fit into a field. Still, if the rank of the matrix happens to be 1023 or less we can overcome this problem somewhat by loading the real components of the column in one field and the imaginary components in another and proceeding with the arithmetic operations accordingly, though this results in degradation of performance over the case where the rank of the matrix is 511 or less.

Case A:

This case is concerned with matrices in which  $N < \frac{WA}{2}$ . We define  $F_j$  as field  $j$  of the AP where  $j$  is an integer. In this report  $F_j$  will normally be 32 bits in length so that there will be eight possible fields or  $j = 1, \dots, 8$ . Let  $F_{j_m}$  denote word  $m$  of field  $j$  where  $0 \leq m \leq WA - 1$ . It should be noted that in some places in the algorithms that follow,  $m = n$  and  $n$  is used for convenience. We need a method of referring to columns of data  $N + 1$  words long. We will utilize the index  $k$  to denote a column of data. Thus,  $F_j^k$  denotes words  $(k-1) \cdot (N+1)$  thru  $[(k-1) \cdot (N+1) + N]$  of field  $j$ . (i.e. the  $N + 1$  words of field  $j$  starting with word  $(k-1) \cdot (N+1)$ ). Let RM denote the word mask register and CR the common register. Also, let ' $\leftarrow$ ' denote the assignment statement, such that the argument on the right of the arrow is assigned to be the content of the argument on the left of the arrow; let ' $\Leftarrow$ ' represent the same meaning as ' $\leftarrow$ ' except that the assignment operation involves loading the arrays from the mass memory; and let ' $\Rightarrow$ ' denote an output operation from the arrays to the mass memory.

The word mask register can be thought of as a 1-bit wide field. With this in mind, we can define  $RM_m$  and  $RM^k$  to have the same meaning as in the case of fields.

We shall also make the assumption that we have a total of at most seven working fields available to us (This would correspond to the STARAN array memory when 32-bit fields are used; the STARAN requires one field to be left aside for use by the machine).

The algorithm for inverting matrices belonging to class A on an AP follows. Memory maps of the AP at various stages of execution of the algorithm are shown in Figure 2. The circled letters within the memory maps are used as keys to the corresponding sets of statements within the algorithm.

#### Algorithm 1

Comment: In A, columns of data are loaded into field F1 of the arrays

A { 0.  $RM \leftarrow 1$  (All words in the array are active)  
1.  $n \leftarrow 1$   
2.  $k \leftarrow 1$   
(n and k here are counters (registers) available to the control unit).  
3.  $F1^k \leftarrow \vec{x}_n$   
4.  $k \leftarrow k + 1$   
5.  $F1^k \leftarrow \vec{y}_n$   
6.  $k \leftarrow k + 1$   
7. Go to ③ if ( $k < M$ )

Fields	0	F1	F2	F3	F4	F5	F6	F7
Words	0	(A) $\overset{\rightarrow}{x}_n$	(B) $\overset{\rightarrow}{y}_n$		(C) $\overset{\rightarrow}{x}_n \cdot F1_n$	(D) $\overset{\rightarrow}{y}_n \cdot F2_n$		
N		$\overset{\rightarrow}{y}_n$	$\overset{\rightarrow}{x}_n$		$\overset{\rightarrow}{y}_n \cdot F1_n$	$\overset{\rightarrow}{x}_n \cdot F2_n$		
$2N + 1$		$\overset{\rightarrow}{x}_n$	$\overset{\rightarrow}{y}_n$		$\overset{\rightarrow}{x}_n \cdot F1_n$	$\overset{\rightarrow}{y}_n \cdot F2_n$		
$3N + 2$		$\overset{\rightarrow}{y}_n$	$\overset{\rightarrow}{x}_n$		$\overset{\rightarrow}{y}_n \cdot F1_n$	$\overset{\rightarrow}{x}_n \cdot F2_n$		
$4N + 3$					$\overset{\rightarrow}{y}_n \cdot F1_n$	$\overset{\rightarrow}{x}_n \cdot F2_n$		
$WA - 1$								

Figure 2 - AP Memory Maps, Case A

	F1	F2	F3	F4	F5	F6	F7
0	$\textcircled{G}$ $\text{Re}(\vec{z}_n \cdot \vec{z}_{n+1})$ = New $\vec{x}_n$	$\vec{y}_n$		$\vec{x}_n \cdot \vec{F1}_n$	$\vec{y}_n \cdot \vec{F2}_n$	$\vec{x}_n \cdot \vec{F1}_n$ + $\vec{y}_n \cdot \vec{F2}_n$	
N	$\text{Im}(\vec{z}_n \cdot \vec{z}_{n+1})$ = New $\vec{y}_n$	$\vec{x}_n$		$\vec{y}_n \cdot \vec{F1}_n$	$-(\vec{x}_n \cdot \vec{F2}_n)$ $-(\vec{x}_n \cdot \vec{F2}_n)$	$\vec{y}_n \cdot \vec{F1}_n$ - $(\vec{x}_n \cdot \vec{F2}_n)$	
2N + 1	$\text{Re}(\vec{z}_n \cdot \vec{z}_{n+1})$ = New $\vec{x}_n$	$\vec{y}_n$		$\vec{x}_n \cdot \vec{F1}_n$	$\vec{y}_n \cdot \vec{F2}_n$	$\vec{x}_n \cdot \vec{F1}_n$ + $\vec{y}_n \cdot \vec{F2}_n$	
25	$\text{Im}(\vec{z}_n \cdot \vec{z}_{n+1})$ = New $\vec{y}_n$	$\vec{x}_n$		$\vec{y}_n \cdot \vec{F1}_n$	$-(\vec{x}_n \cdot \vec{F2}_n)$ $-(\vec{x}_n \cdot \vec{F2}_n)$	$\vec{y}_n \cdot \vec{F1}_n$ - $(\vec{x}_n \cdot \vec{F2}_n)$	
3N + 3							
4N + 3							
WA - 1							

Figure 2. - Continued

	F1	F2	F3	F4	F5	F6	F7
0	New $\vec{x}_n$	(ii) New $\vec{y}_n$	(i) $\vec{x}_1$	(j) $\vec{x}_n \cdot \vec{x}_{1n}$	(k) $-\vec{y}_n \cdot \vec{y}_{1n}$	(l) $\vec{x}_n \cdot \vec{x}_{1n}$	(m) $\vec{y}_3 - \vec{y}_6$
N	New $\vec{y}_n$	New $\vec{x}_n$	$\vec{y}_1$	$\vec{y}_n \cdot \vec{x}_{1n}$	$\vec{x}_n \cdot \vec{y}_{1n}$	$-\vec{y}_n \cdot \vec{y}_{1n}$	New $\vec{x}_1$
2N + 1	New $\vec{x}_n$	New $\vec{y}_n$	$\vec{x}_2$	$\vec{x}_n \cdot \vec{x}_{2n}$	$-\vec{y}_n \cdot \vec{y}_{2n}$	$\vec{x}_n \cdot \vec{x}_{2n}$	New $\vec{x}_2$
26	New $\vec{y}_n$	New $\vec{x}_n$	$\vec{y}_2$	$\vec{y}_n \cdot \vec{x}_{2n}$	$-\vec{y}_n \cdot \vec{y}_{2n}$	$-\vec{y}_n \cdot \vec{y}_{2n}$	
3N + 2	New $\vec{y}_n$	New $\vec{x}_n$					
4N + 3							
WA - 1							
					$\vec{x}_1, \vec{x}_2 \neq \vec{x}_n$	$\vec{y}_1, \vec{y}_2 \neq \vec{y}_n$	

Figure 2. - Continued

Comment: In B, columns of data are loaded into field F2 of the arrays  
but with  $\vec{y}_n$  above  $\vec{x}_n$ .

B      {  
8.     $k \leftarrow 1$   
9.     $F2^k \leftarrow \vec{y}_n$   
10.    $k \leftarrow k + 1$   
11.    $F2^k \leftarrow \vec{x}_n$   
12.    $k \leftarrow k + 1$   
13.   Go to (9) if ( $k < M$ )

Comment: The steps in C cause the following to be performed: multiply  
the data in F1 by the pivot element of the real parts of the  
complex numbers and place the result in F4.

C      {  
14.    $CR \leftarrow F1_n$   
15.    $F4 \leftarrow F1 \times CR$

Comment: The steps in D cause the following to be performed: multiply the  
data in F2 by the pivot element of the imaginary parts of the  
complex numbers and place the result in F5.

D      {  
16.    $CR \leftarrow F2_n$   
17.    $F5 \leftarrow F2 \times CR$

Comment: The steps in E and F cause the following to be performed: multiply the reals by a minus and add F4 and F5.

E      {  
18.  $RM^{1,3,\dots,k-2} \leftarrow 0$   
19.  $F5 \leftarrow -F5$   
  
F      {  
20.  $RM \leftarrow 1$   
21.  $F6 \leftarrow F4 + F5$

Comment: The steps in G complete the normalization of the pivot element to one.

G      {  
22.  $CR \leftarrow F6_n$   
23.  $F1 \leftarrow F6 + CR$   
24.  $F6 \leftarrow F1$   
25.  $CR \leftarrow F6_n$   
26.  $CR \leftarrow CR - 1$   
27.  $F6_n \leftarrow CR$

Comment: The  $n^{\text{th}}$  word of field 6, which has just been reduced to 1, is now replaced with a zero, prior to loading the contents of the field on to the mass memory because the corresponding matrix column will no longer act as a pivot in future iterations.

28.  $\xrightarrow{n} F6^1$

29.  $\xrightarrow{n} F6^2$

Comment: The  $n$  on top of ' $\Rightarrow$ ' implies that the vectors being read out correspond to the  $n^{th}$  column of the matrix.

Comment: The steps in H place the new  $\vec{y}_n$  and  $\vec{x}_n$  in  $F2$  with  $\vec{y}_n$  on top.

(These are the same vectors that were just previously read out.

Doing this operation in this way conserves time.)

- H {
30.  $k \leftarrow 1$
  31.  $F2^k \leftarrow \vec{y}_n$
  32.  $k \leftarrow k + 1$
  33.  $F2^k \leftarrow \vec{x}_n$
  34.  $k \leftarrow k + 1$
  35. Go to 31 if ( $k < M$ )
  36.  $n1 \leftarrow 1$
  37.  $n2 \leftarrow 1$
- ( $n1$  and  $n2$  are counters)

**Comment:** The steps in I bring in additional columns of the matrix to F3.

In Figure 2 we show the first two columns of the matrix.

Eventually, all columns would be brought in with the exception  
of the  $n^{\text{th}}$  column.

- I
- { 38.  $k \leftarrow 1$
  - 39. Go to 44 if ( $n_1 = n$ )
  - 40.  $F3^k \leftarrow \vec{x}_{n_1}$
  - 41.  $k \leftarrow k + 1$
  - 42.  $F3^k \leftarrow \vec{y}_{n_1}$
  - 43.  $k \leftarrow k + 1$
  - 44.  $n_1 \leftarrow n_1 + 1$
  - 45. Go to 39 if  $\{k < M\} \text{ and } \{n_1 < N\}$
  - 46.  $\ell \leftarrow 1$
  - 47.  $RM \leftarrow 0$  except  $RM^{\ell, \ell+1} \leftarrow 1$   
( $\ell$  is a counter)

**Comment:** In steps J and K the columns are multiplied by the proper value(s)  
in preparation for the subtraction process.

- J
- { 48.  $CR \leftarrow F3_{(\ell-1)(N+1) + n}$
  - 49.  $F4 \leftarrow F1 \times CR$

Comment: The steps in I bring in additional columns of the matrix to F3.

In Figure 2 we show the first two columns of the matrix.

Eventually, all columns would be brought in with the exception  
of the  $n^{\text{th}}$  column.

- I
- {
38.  $k \leftarrow 1$
  39. Go to 44 if ( $n_1 = n$ )
  40.  $F3^k \leftarrow \vec{x}_{n_1}$
  41.  $k \leftarrow k + 1$
  42.  $F3^k \leftarrow \vec{y}_{n_1}$
  43.  $k \leftarrow k + 1$
  44.  $n_1 \leftarrow n_1 + 1$
  45. Go to 39 if  $\{k < M\} \text{ and } \{n_1 < N\}$
  46.  $\ell \leftarrow 1$
  47.  $RM \leftarrow 0 \text{ except } RM^\ell, \ell + 1 \leftarrow 1$   
( $\ell$  is a counter)

Comment: In steps J and K the columns are multiplied by the proper value(s)  
in preparation for the subtraction process.

- J
- {
48.  $CR \leftarrow F3^{(\ell-1)(N+1) + n}$
  49.  $F4 \leftarrow F1 \times CR$

K      {
   
 50.  $CR \leftarrow F3_{(N+1)} + n$ 
  
 51.  $F5 \leftarrow F2 \times CR$ 
  
 52.  $i \leftarrow i + 2$ 
  
 53. Go to (47) if ( $i < k$ )
   
 54.  $RM \leftarrow 1$ 
  
 55.  $RM^{2,4,\dots,k-1} \leftarrow 0$ 
  
 56.  $F5 \leftarrow -F5$

**Comment:** In L and M the processing of  $\vec{x}_1, \vec{y}_1$  and  $\vec{x}_2, \vec{y}_2$  is completed.

The rest of the algorithm deals with processing the rest of the columns and stopping conditions.

L      {
   
 57.  $RM \leftarrow 1$ 
  
 58.  $F6 \leftarrow F4 + F5$ 
  
 M      {
   
 59.  $F5 \leftarrow F3 - F6$ 
  
 60.  $i \leftarrow 1$ 
  
 61. Go to (65) if ( $n_2 \neq n$ )
   
 62.  $CR \leftarrow F5_{(i-1) \cdot (N+1)} + n$ 
  
 63.  $CR \leftarrow CR + 1$ 
  
 64.  $F5_{(i-1) \cdot (N+1)} + n \leftarrow CR$ 
  
 65.  $\overline{F5^i} \leftarrow F5^i$ 
  
 66.  $i \leftarrow i + 1$

67.  $\xrightarrow{n_2} F5^l$   
 68.  $l \leftarrow l + 1$   
 69.  $n_2 \leftarrow n_2 + 1$   
 70. Go to (61) if ( $l < k$ ) and ( $n_2 \leq N$ )  
 71. Go to (38) if ( $n_1 \leq N$ )  
 72.  $n \leftarrow n + 1$   
 73. Go to (2) if ( $n \leq N$ )  
 74. STOP

We can now use the matrix A of Figure 1 to illustrate how algorithm 1 would be used to invert this matrix on an AP. The  $\vec{x}_n$  and  $\vec{y}_n$  vectors,  $n = 1, \dots, N$ , corresponding to A are shown in Figure 3. Shown in Figures 4.1 to 4.4 are the AP memory maps that would result during a partial inversion of the above matrix using algorithm 1. Circled letters within the Figures are again used as keys to the corresponding sets of statements within the algorithm.

Initially, vectors  $\vec{x}_1$  and  $\vec{y}_1$  are loaded into fields F1 and F2 of the AP as shown in Figure 4.1. This corresponds to the first column of the matrix as being the pivot. The process of division of this pivot by the first diagonal element of the matrix is contained in Figures 4.1 and 4.2.

$$z = \begin{bmatrix} 1 + i0 & 0 + i0 & 0 + i0 \\ 1 + i1 & -2 - i2 & 3 + i3 \\ 0 + i0 & 2 + i2 & -3 - i3 \\ 1 + i1 & 1 + i1 & 1 + i1 \end{bmatrix}$$

↑                   ↑                   ↑

$\vec{z}_1$                 $\vec{z}_2$                 $\vec{z}_3$

Correspondingly,

$$\begin{aligned}\vec{x}_1 &= \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}, & \vec{y}_1 &= \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}, \\ \vec{x}_2 &= \begin{bmatrix} 0 \\ -2 \\ 2 \\ 1 \end{bmatrix}, & \vec{y}_2 &= \begin{bmatrix} 0 \\ -2 \\ 2 \\ 1 \end{bmatrix}, \\ \vec{x}_3 &= \begin{bmatrix} 0 \\ 3 \\ -3 \\ 1 \end{bmatrix}, & \text{and} & \vec{y}_3 &= \begin{bmatrix} 0 \\ 3 \\ -3 \\ 1 \end{bmatrix}.\end{aligned}$$

FIGURE 3:  $\vec{x}_n$  and  $\vec{y}_n$  Vectors Corresponding to Matrix A of Figure 1.

Word	F1	F2	F3	F4	F5	F6	F7
0	(A) 1	(B) 0		(C) 1	(D) 0		
1	1	1		1	1		
2	0	0		0	0		
3	1	1		1	1		
4	0	1		0	1		
5	1	1		1	1		
6	0	0		0	0		
7	1	1		1	1		
8	1	0		1	0		
9	1	1		1	1		
10	0	0		0	0		
11	1	1		1	1		
12	0	1		0	1		
13	1	1		1	1		
14	0	0		0	0		
15	1	1		1	1		
16							
17							

$CR \leftarrow F1_1 ; F4 \leftarrow F1 \times CR$

$CR \leftarrow F2_1 ; F5 \leftarrow F2 \times CR$

FIGURE 4.1: Initial AP Memory Map During Inversion of the A Matrix  
Using Algorithm 1.

Word	Field $\rightarrow$	F1	F2	F3	F4	F5	F6	F7
k = 1	0	(G) 0.5			1	(E) 0	(F) 1	
	1	1			1	1	2	
	2	0			0	0	0	
	3	1			1	1	2	
k = 2	4	-0.5			0	-1	-1	
	5	0			1	-1	0	
	6	0			0	0	0	
	7	0			1	-1	0	
	8	0.5			1	0	1	
k = 3	9	1			1	1	2	
	10	0			0	0	0	
	11	1			1	1	2	
	12	-0.5			0	-1	-1	
k = 4	13	0			1	-1	0	
	14	0			0	0	0	
	15	0			1	-1	0	
	16							
	17							

Mask out words 0 thru 3 and words 8 thru 11 of F5;  $F5 \leftarrow -F5$

i.e.,  $F5'$

i.e.,  $F5^3$

$$F6 \leftarrow F4 + F5$$

$$CR \leftarrow F6_1; F1 \leftarrow F6 + CR$$

$$P1^1 = \text{New } \vec{x}_1; P1^2 = \text{New } \vec{y}_1.$$

FIGURE 4.2: Intermediate AP Memory Map.

Word	Field $\Rightarrow$	F1	F2	F3	F4	F5	F6	F7
$k = 1$	0	0.5	(H) -0.5	(I) 0	(J) -1	(K) 1		
	1	1	0	-2	-2	0		
	2	0	0	2	0	0		
	3	1	0	1	-2	0		
$k = 2$	4	-0.5	0.5	0	1	-1		
	5	0	1	-2	0	-2		
	6	0	0	2	0	0		
	7	0	1	1	0	-2		
$k = 3$	8	0.5	-0.5	0	1.5	-1.5		
	9	1	0	3	3	0		
	10	0	0	-3	0	0		
	11	1	0	1	3	0		
$k = 4$	12	-0.5	0.5	0	-1.5	1.5		
	13	0	1	3	0	.3		
	14	0	0	-3	0	0		
	15	0	1	1	0	3		
16								
17								

$F2^1 \leftarrow \text{New } \vec{y}_1; F2^2 \leftarrow \text{New } \vec{x}_1; F2^3 \leftarrow \text{New } \vec{y}_1; F2^4 \leftarrow \text{New } \vec{x}_1$

$F3^1 \leftarrow \vec{x}_2; F3^2 \leftarrow \vec{y}_2; F3^3 \leftarrow \vec{x}_3; F3^4 \leftarrow \vec{y}_3$

Mask out  $F3^3$  and  $F3^4$ ;  $CR \leftarrow F3_1$ ;  $F4 \leftarrow F1 \times CR$ ;

$CR \leftarrow F3_5$ ;  $F5 \leftarrow F2 \times CR$

Mask out  $F3^1$  and  $F3^2$ ;  $CR \leftarrow F3_9$ ;  $F4 \leftarrow F1 \times CR$

$CR \leftarrow F3_{13}$ ;  $F5 \leftarrow F2 \times CR$

FIGURE 4.3: Intermediate AP Memory Map

Field →	$F_1$	$F_2$	$F_3$	$F_4$	$F_5$	$F_6$	$F_7$
Word	0				(K)	(L)	(M)
$k = 1$	1		-2	-2	0	-2	0
	2		2	0	0	0	2
	3		1	-2	0	-2	3
	4		0	1	-1	0	0
$k = 2$	5		-2	0	-2	-2	0
	6		2	0	0	0	2
	7		1	0	-2	-2	3
	8		0	1.5	1.5	3	-3
$k = 3$	9		3	3	0	3	0
	10		-3	0	0	0	-3
	11		1	3	0	3	-2
	12		0	-1.5	1.5	0	0
$k = 4$	13		3	0	3	3	0
	14		-3	0	0	0	-3
	15		1	0	3	3	-2
	16						
17							

Mask out  $F_5^2$  and  $F_5^4$ ;  $F_5 \leftarrow -F_5$

$F_6 \leftarrow F_4 + F_5$

$F_7 \leftarrow F_3 - F_6$

$F_7^1 = \text{New } \vec{x}_2; F_7^2 = \text{New } \vec{y}_2; F_7^3 = \text{New } \vec{x}_3; F_7^4 = \text{New } \vec{y}_3$

FIGURE 4.4: Intermediate AP Memory Map.

Once the pivot has been so 'normalized', columns 2 and 3 of the matrix would be ready to be processed against it and as such these columns are brought into the AP (in field F3) as shown in Figure 4.3. The processing of columns 2 and 3 of the matrix against the pivot is shown in Figures 4.3 and 4.4.

The rest of the inversion process would be identical to that shown in Figures 4.1 to 4.4, except that we would next use column 2 of the matrix as the pivot, 'normalize it', process the rest of the matrix columns against it, and finally do the same with column 3.

Case B

Matrices with  $\frac{WA}{2} \leq N \leq (WA - 1)$ .

When the rank of the matrix is greater than or equal to  $\frac{WA}{2}$ , a complete column of the matrix (both the  $\vec{x}_n$  and  $\vec{y}_n$  vectors) can no longer be fit into one field of the AP. For these matrices, the inversion algorithm has thus to be revised accordingly. Note that since  $N$  is less than  $WA$ , we can still go about our business, albeit somewhat inefficiently, by splitting the matrix column in two fields: the real part in one field and the imaginary part in another. However, for matrices with rank greater than or equal to  $WA$ , there would be a much greater than proportional decrease in efficiency since a matrix column would have to be spread over more than two fields of the array, i.e., it would have to be processed in parts.

The algorithm for inverting case B matrices on an AP follows. The memory maps at various stages of execution of the algorithm are illustrated in Figure 5. Keys are provided as before from Figure 5 to the corresponding steps in the algorithm.

Algorithm 2

0.  $RM \leftarrow 1$
1.  $n \leftarrow 1$
- A { (2.)  $F1 \leftarrow \vec{x}_n$
- B { 3.  $F2 \leftarrow \vec{y}_n$
- C { 4.  $CR \leftarrow F1_n$   
5.  $F3 \leftarrow F1 \times CR$

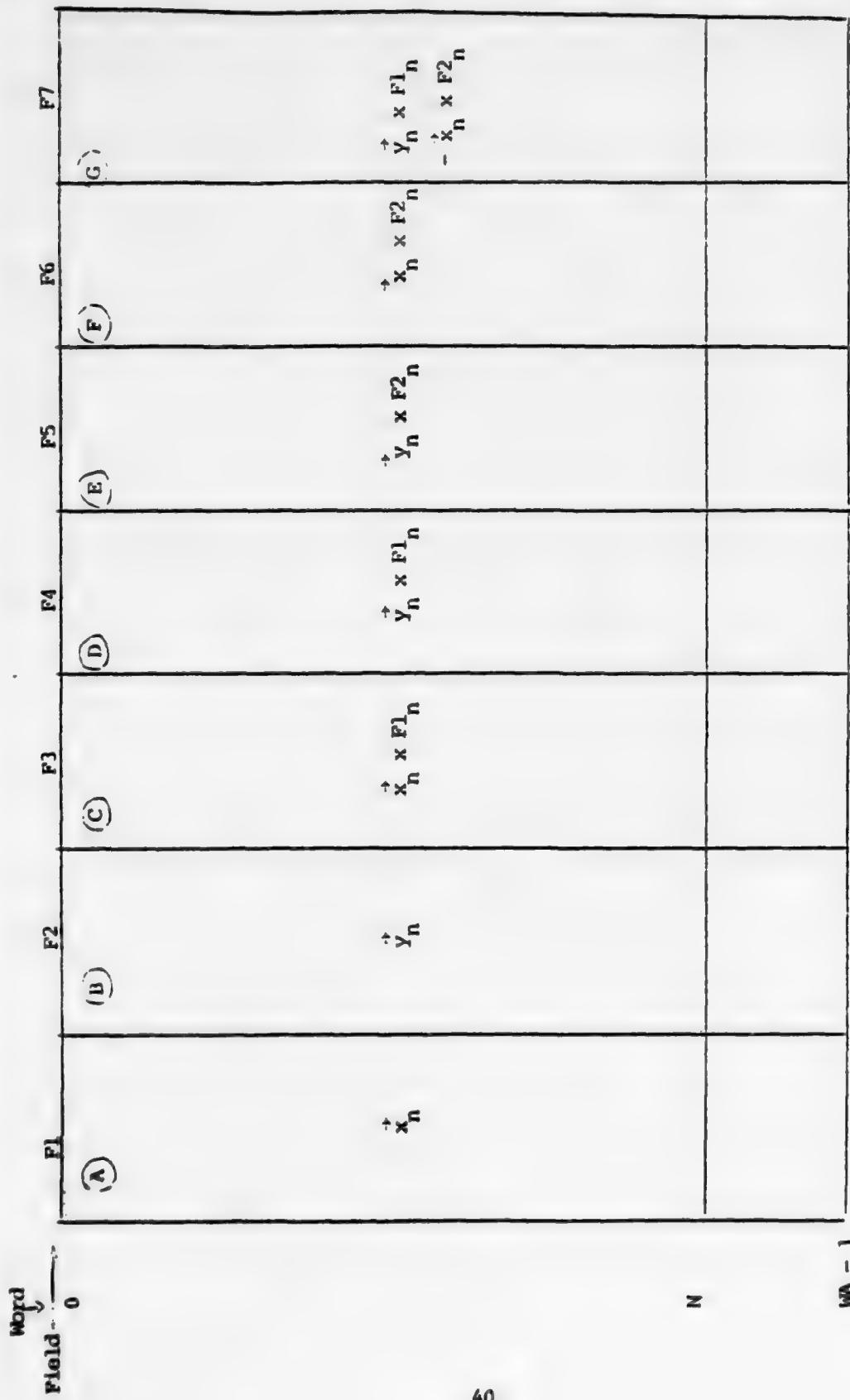


Figure 5 - AP Memory Maps, Case B

F7	F6	F5	F4
F3	F2	F1	0
(I)	(J)	(H)	
$\text{Re}(\vec{z}_n + \vec{z}'_n) = \text{New } \vec{x}_n$	$\text{Im}(\vec{z}_n + \vec{z}'_n) = \text{New } \vec{y}_n$	$\vec{x}_n \times F1_n$ $\vec{y}_n \times F2_n$	

Figure 5. - Continued

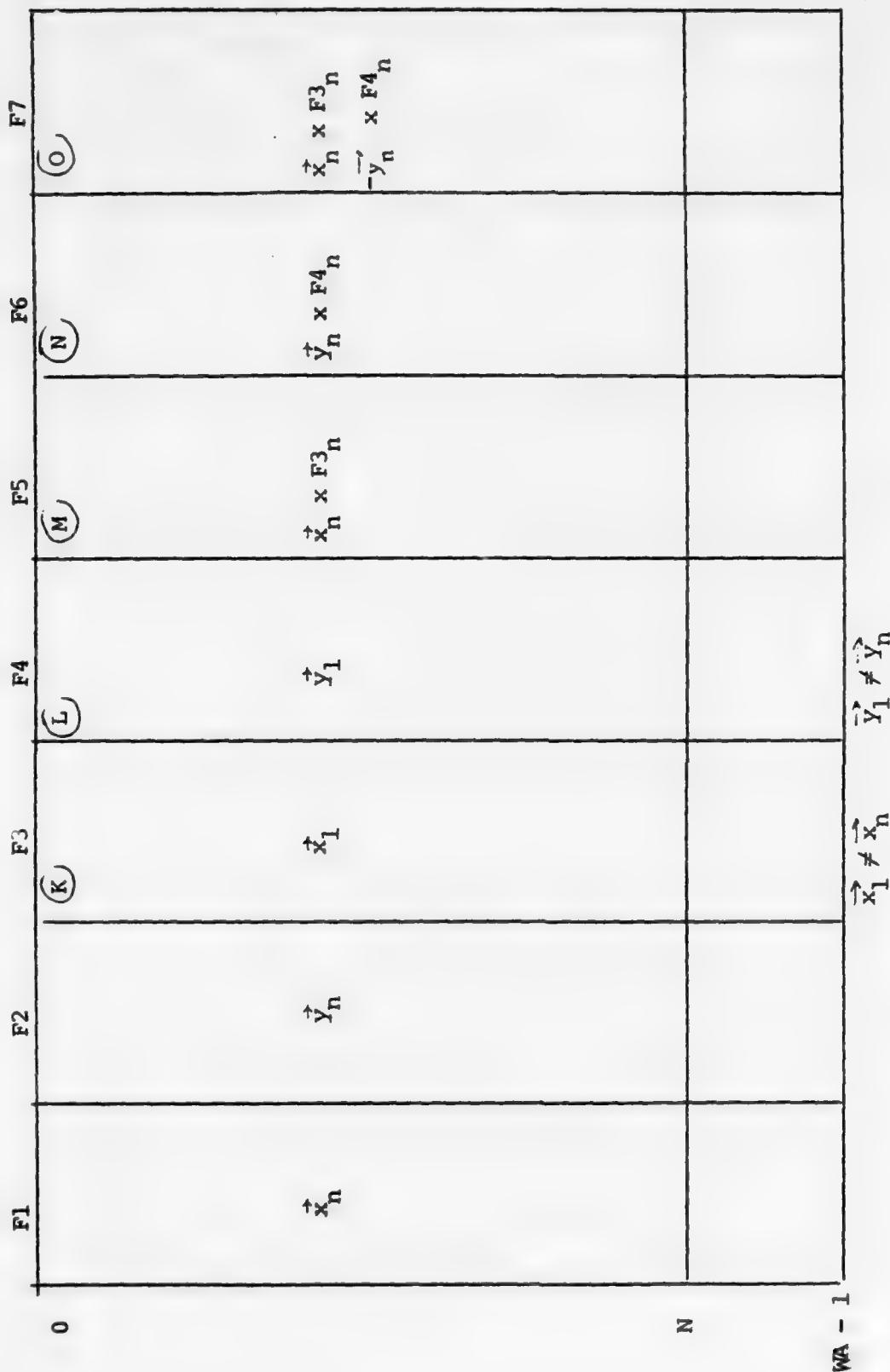


Figure 5. - Continued

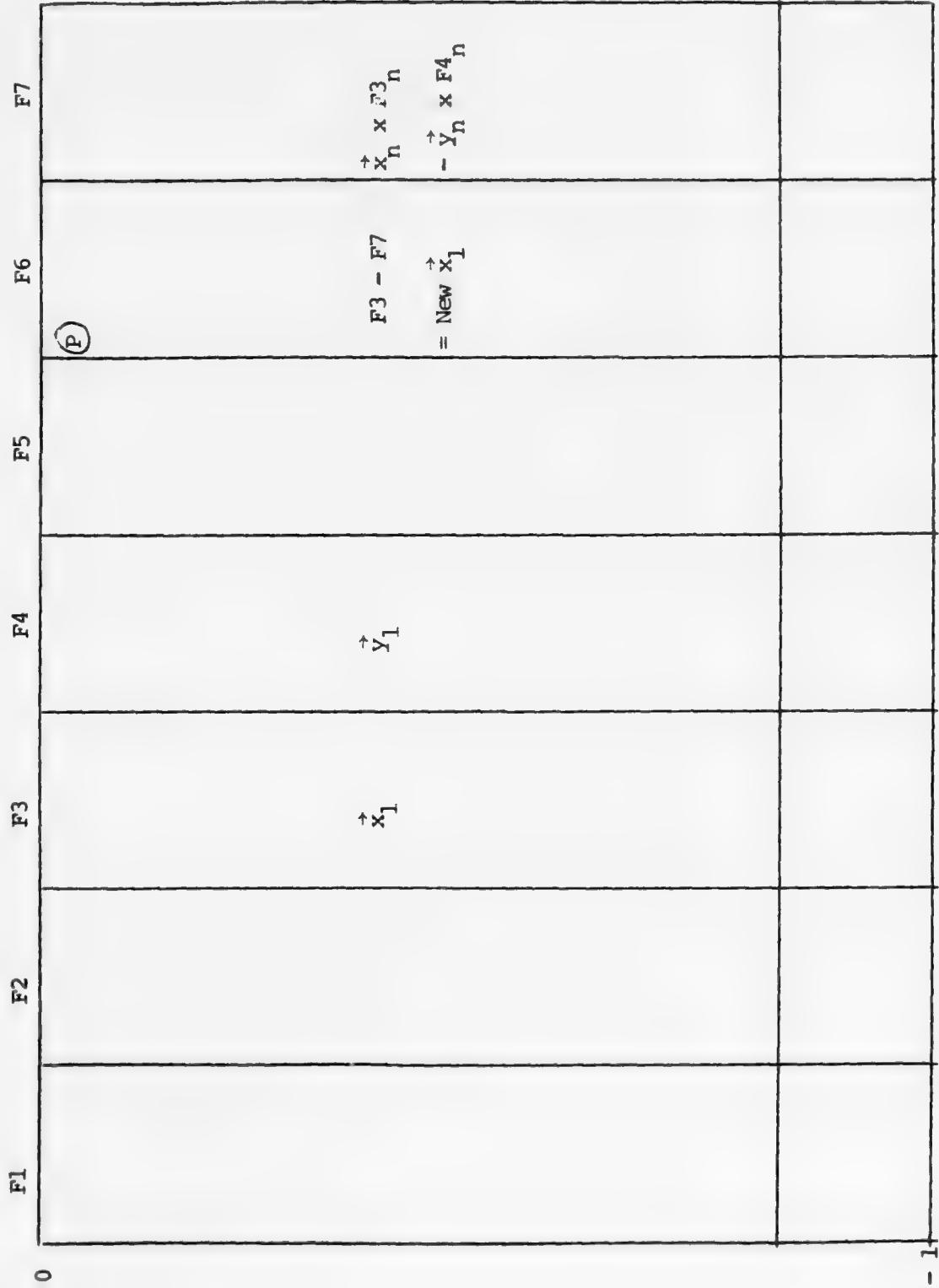


Figure 5. - Continued

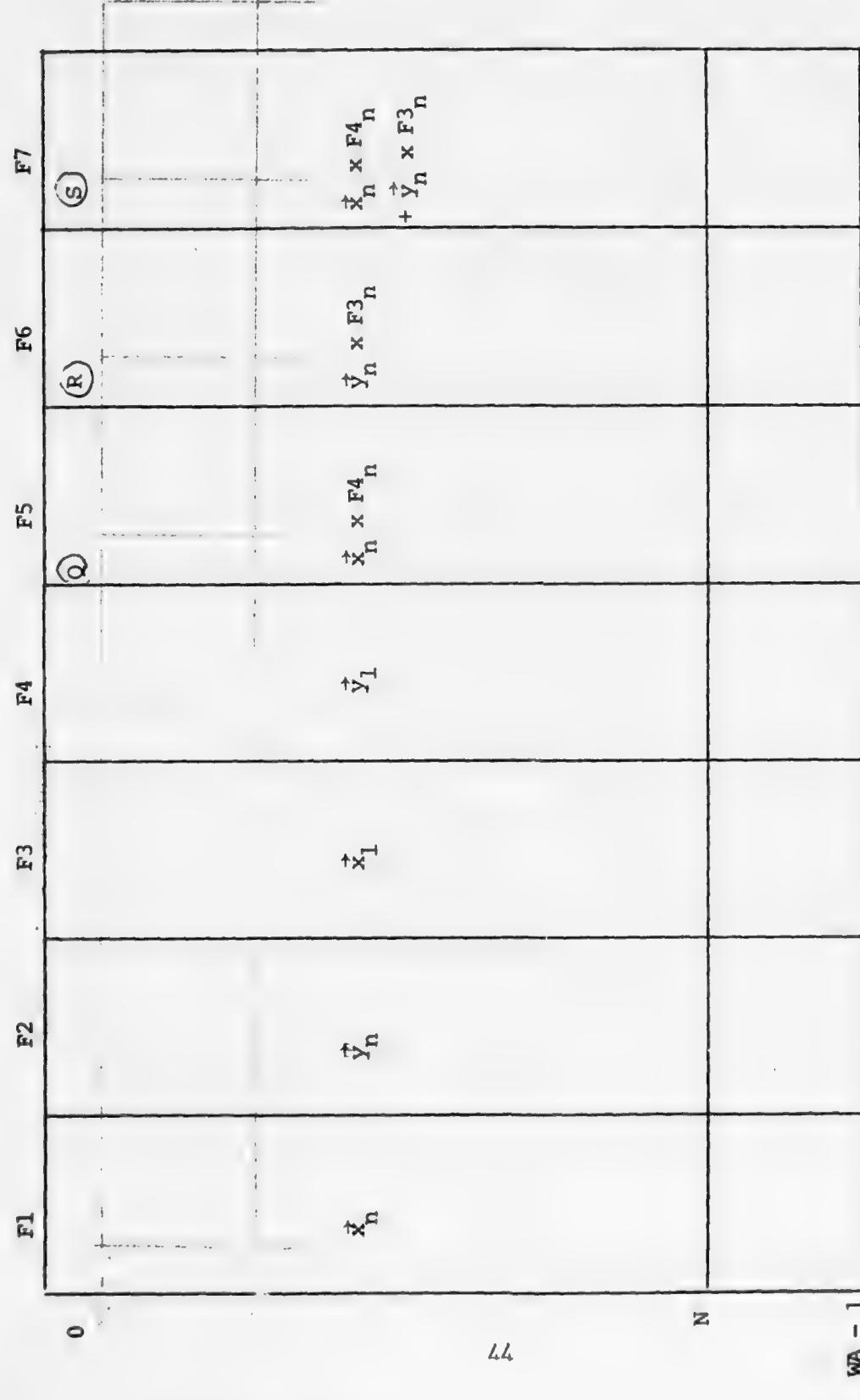


Figure 5. - Continued

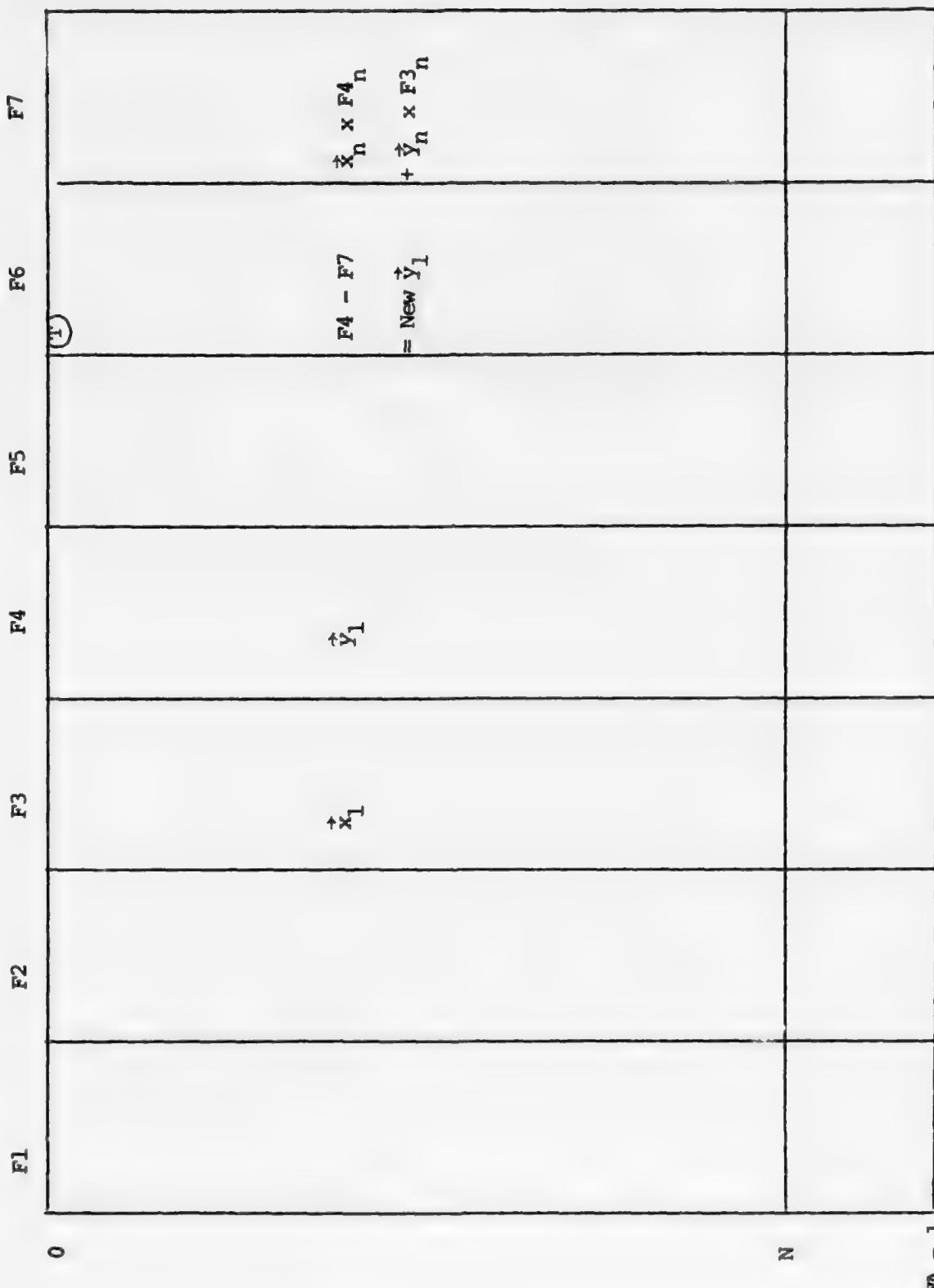


Figure 5. - Continued

- D 6.  $F4 \leftarrow F2 \times CR$   
7.  $CR \leftarrow F2_n$   
E 8.  $F5 \leftarrow F2 \times CR$   
F 9.  $F6 \leftarrow F1 \times CR$   
G 10.  $F7 \leftarrow F4 - F6$   
H 11.  $F6 \leftarrow F3 + F5$   
I 12.  $CR \leftarrow F6_n$   
J 13.  $F1 \leftarrow F6 + CR$   
14.  $F2 \leftarrow F7 + CR$   
15.  $F7 \leftarrow F1$   
16.  $CR \leftarrow F7_n$   
17.  $CR \leftarrow CR - 1$   
18.  $F7_n \leftarrow CR$   
19.  $\xrightarrow{n} F7$   
20.  $\xrightarrow{n} F2$   
21.  $n1 \leftarrow 1$   
22. Go to ④ if ( $n1 = n$ )  
K 23.  $F3 \leftarrow \vec{x}_{n1}$   
L 24.  $F4 \leftarrow \vec{y}_{n1}$   
M 25.  $CR \leftarrow F3_n$   
26.  $F5 \leftarrow F1 \times CR$   
N 27.  $CR \leftarrow F4_n$   
28.  $F6 \leftarrow F2 \times CR$

- O 29.  $F7 \leftarrow F5 - F6$
- P 30.  $F6 \leftarrow F3 - F7$
31. Go to (35) if  $\{nl \neq (n+1)\}$
32.  $CR \leftarrow F6_n$
33.  $CR \leftarrow CR + 1$
34.  $F6_n \leftarrow CR$
- (35.)  $\xrightarrow{nl} F6$
- Q. 36.  $F5 \leftarrow F1 \times CR$
- R 37.  $CR \leftarrow F3_n$
38.  $F6 \leftarrow F2 \times CR$
- S 39.  $F7 \leftarrow F5 + F6$
- T 40.  $F6 \leftarrow F4 - F7$
41.  $\xrightarrow{nl} F6$
- (42.)  $nl \leftarrow nl + 1$
43. Go to (22) if  $(nl \leq N)$
44.  $n \leftarrow n + 1$
45. Go to (2) if  $(n \leq N)$
46. STOP.

After complete execution of the algorithm we would end up with a  $(N+1) \times N$  matrix in mass storage, the first  $N$  rows of which are the desired inverse (the last row would correspond to the last row of the identity matrix). The algorithm described above requires the fewest number of steps of all the various other possibilities that were considered.

### Development of Timing Expressions

In the development of the timing expressions we shall consider all the AP operations involved in the algorithms viz: arithmetic operations and setting of the mask register and common register, as well as array loading and unloading times. We shall, however, neglect the times to search for and prepare the data in mass memory prior to loading. We shall also neglect the times for setting counters in the AP control unit (as in steps 1 and 2 of the algorithm), which are negligible compared to the other times.

Shown in Table 2 are memory and arithmetic operation times for the various operations that are used in the previously developed algorithms. These times are based on the RADC STARAN computer.

The timing expression for executing the inversion algorithm can be derived by adding the times required for the individual steps, in the algorithm, of course taking into account all the loops that exist. This leads to the following expression for Case A (M has the same meaning as defined on page 20):

$$\begin{aligned} \text{Time}_A = & N [ M \times 32 \times t_1 + M \times 32 \times t_1 \\ & + t_{lc} + t_m + t_{lc} + t_m + t_{rm} + t_{nm} + t_{rm} + t_a + t_{lc} + t_d \\ & + t_{nc} + t_{lc} + t_{crd} + t_{wc} + 2 \times 32 \times t_1 \\ & + M \times 32 \times t_1 \\ & + 2(N-1) \times 32 \times t_1 \\ & + (N-1) (t_{rm} + t_{lc} + t_m + t_{lc} + t_m) + K(t_{rm} + t_{nm} + t_{rm} + t_a + t_s) \\ & + 2(N-1) \times 32 \times t_1 \\ & + t_{lc} + t_{cri} + t_{wc} ], \end{aligned}$$

TABLE 2: Memory and Arithmetic Operation Times for STARAN

OPERATION	Times*
$t_l$ = time to load a bit-slice	0.3 usecs.
$t_{lc}$ = time to load CR from AM word	.64 usecs.
$t_{rm}$ = time to set M response store	.26 usecs.
$t_m$ = time to multiply field by CR - FL - SP	715 usecs.
$t_d$ = time to divide field by CR - FL - SP	772 usecs.
$t_a$ = time to add field to field - FL - SP	412 usecs.
$t_{rn}$ = time to move neg of a field	.66n + 3.74usecs
$t_s$ = time to subtract field from field - FL - SP	412 usecs.
$t_{nc}$ = time to copy a field of width n	.60n + .51usecs.
$t_{crd}$ = time to decrement CR	.25 usecs.
$t_{cri}$ = time to increment CR	.25 usecs.
$t_{wc}$ = time to store CR into AM word	.86 usecs.

\*n denotes the field width; for purposes of this analysis, n = 32.

Which simplifies to:

$$\begin{aligned} \text{Time}_A = N[ & (3M + 2) \times 32t_1 + t_{nc} \\ & + 5t_{lc} + 2t_{rm} + t_{nm} + 2t_m + t_a + t_d + t_{ord} + t_{cri} + 2t_{wc} \\ & + (N-1) (4 \times 32 t_1 + t_{rm} + 2t_{lc} + 2t_m) \\ & + K(2t_{rm} + t_{nm} + t_a + t_s) ]. \end{aligned}$$

For STARAN, the above timing expression reduces to:

$$\text{Time}_A = N[28.8M + 1469.94N + 849.38K + 1213.77] \text{ usecs.}$$

Summing up the times required for the individual steps in the algorithm as was done for Case A, we obtain the following timing expression for inverting matrices belonging to Case B:

$$\begin{aligned} \text{Time}_B = N[ & 2 \times 32t_1 + 2(t_{lc} + 2t_m) + t_s + t_a + t_{lc} + 2t_d + 2 \times 32t_1 \\ & + (N-1) \{ 2 \times 32t_1 + 2(t_{lc} + t_m) + 2t_s + 32t_1 \\ & \quad + t_m + t_{lc} + t_m + t_a + t_s + 32t_1 \} \\ & + t_{nc} + t_{lc} + t_{ord} + t_{wc} + t_{lc} + t_{cri} + t_{wc} ], \end{aligned}$$

which simplifies to:

$$\begin{aligned} \text{Time}_B = N[ & 128t_1 + 5t_{lc} + 4t_m + t_s + t_a + 2t_d + t_{nc} + t_{crd} + t_{cri} + 2t_{wc} \\ & + (N-1) \{ 128t_1 + 3t_{lc} + 4t_m + 3t_s + t_a \} ]. \end{aligned}$$

For STARAN, the above timing expression reduces to:

$$\text{Time}_B = N[4548.32 N + 743.21] \text{ usecs.}$$

Notice that for a 1024 word STARAN, the above timing figure would correspond to matrices with  $512 \leq N \leq 1023$ .

If we solve the two equations ( $\text{Time}_A$  and  $\text{Time}_B$ ) utilizing various values of rank  $N$  we obtain the curves for complex numbers as shown in Figures 6 and 7. The curves for real numbers were taken from reference [1]. In Figure 6 the break in the curve occurs when we can no longer fit two columns (both real and imaginary) of the matrix into the same field of the arrays. In Figure 7 the break occurs at  $N = 512$  when the algorithms change from Case A to Case B. Shown in Tables 3 and 4 are the supporting data values for these curves. Shown in Figure 8 and Table 5 are similar data for 16 arrays.

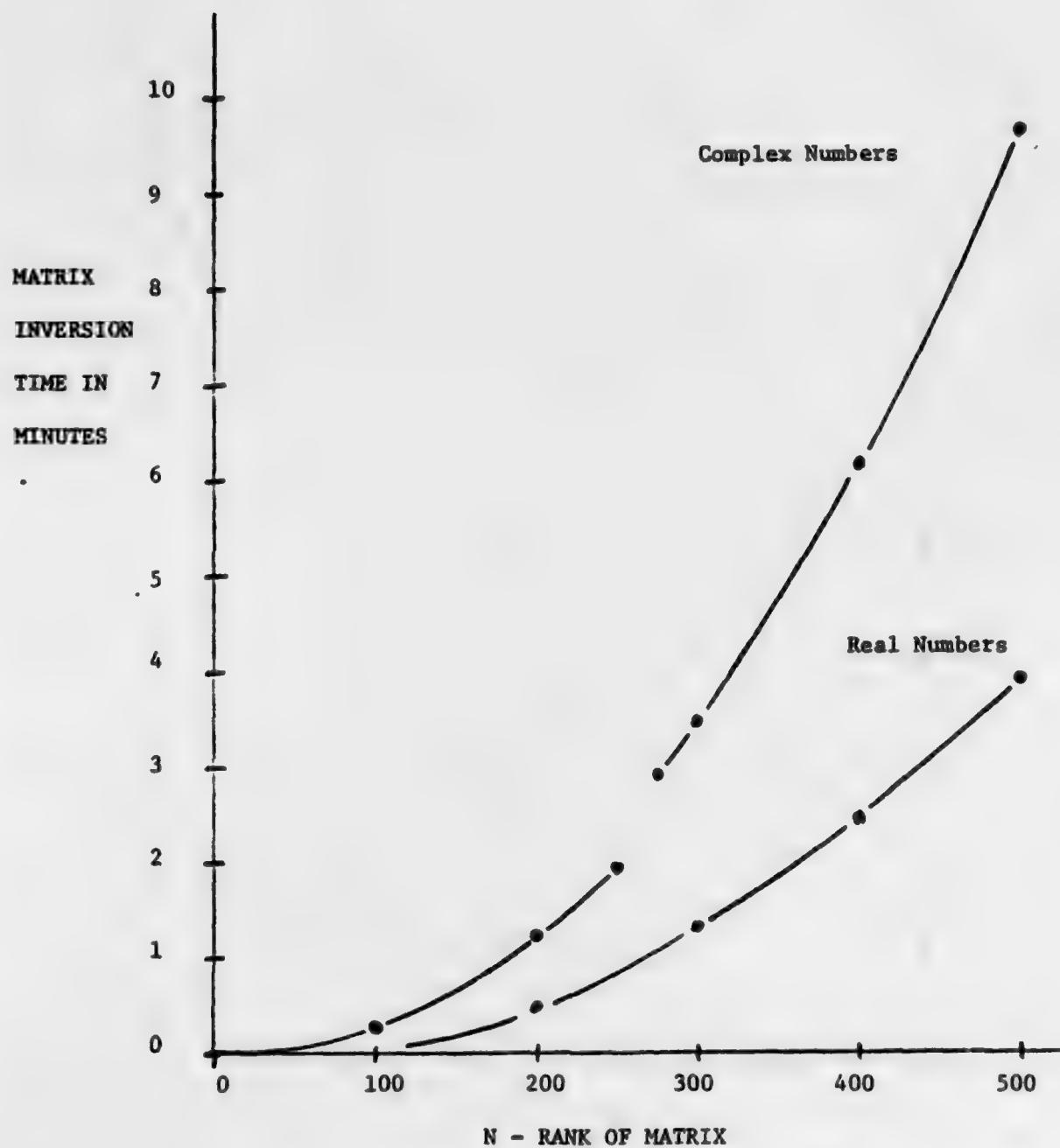


FIGURE 6. Matrix Inversion Time with 4 Arrays and Rank < 511.

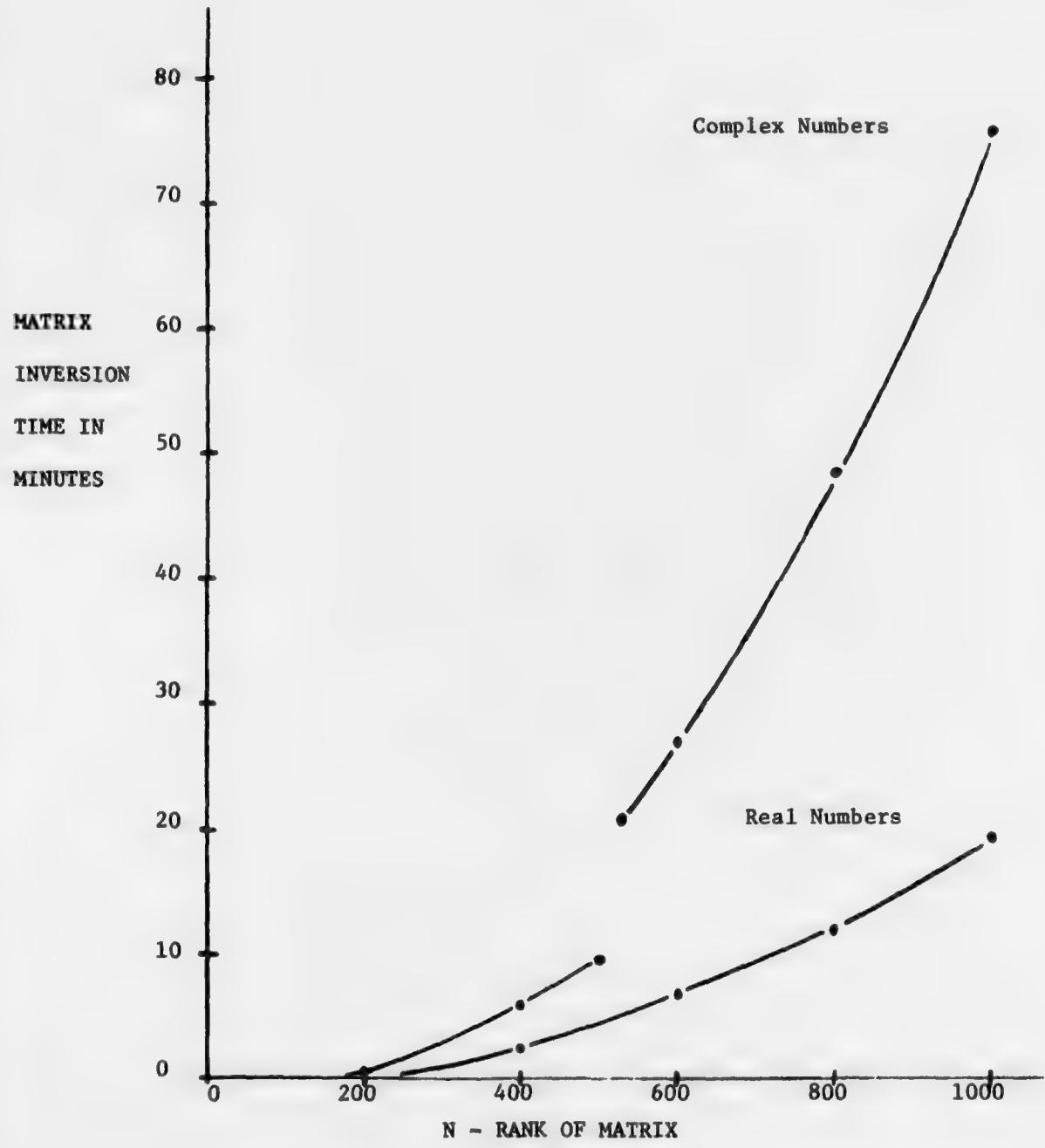


FIGURE 7. Matrix Inversion Time with 4 Arrays and Rank < 1023.

TABLE 3: Timing Data for Inverting a Matrix with Complex  
Numbers for  $25 < \text{Rank} < 511$ . (4 Arrays)

<u>RANK</u>	<u>TIME</u>		
	<u>HRS.</u>	<u>MINS.</u>	<u>SECS.</u>
25	0	0	1
50	0	0	4
75	0	0	9
100	0	0	17
125	0	0	27
150	0	0	40
175	0	0	58
200	0	1	16
225	0	1	36
250	0	1	59
275	0	2	56
300	0	3	29
325	0	4	5
350	0	4	44
375	0	5	26
400	0	6	11
425	0	6	59
450	0	7	50
475	0	8	43
500	0	9	40

TABLE 4: Timing Data for Inverting a Matrix with Complex Numbers  
for  $25 < \text{Rank} < 1023$ . (4 Arrays)

<u>RANK</u>	<u>TIME</u>		
	<u>HRS.</u>	<u>MINS.</u>	<u>SECS.</u>
25	0	0	1
50	0	0	4
75	0	0	9
100	0	0	17
125	0	0	27
150	0	0	40
175	0	0	58
200	0	1	16
225	0	1	36
250	0	1	59
275	0	2	56
300	0	3	29
325	0	4	5
350	0	4	44
375	0	5	26
400	0	6	11
425	0	6	59
450	0	7	50
475	0	8	43
500	0	9	40
511	0	10	6
512	0	19	53
525	0	20	54
550	0	22	56
575	0	25	4
600	0	27	18
625	0	29	37
650	0	32	2
675	0	34	33
700	0	37	9
725	0	39	51
750	0	42	39
775	0	45	32
800	0	48	32
825	0	51	36
850	0	54	48
875	0	58	3
900	1	1	25
925	1	4	52
950	1	8	26
975	1	12	4
1000	1	15	49
1023	1	19	21

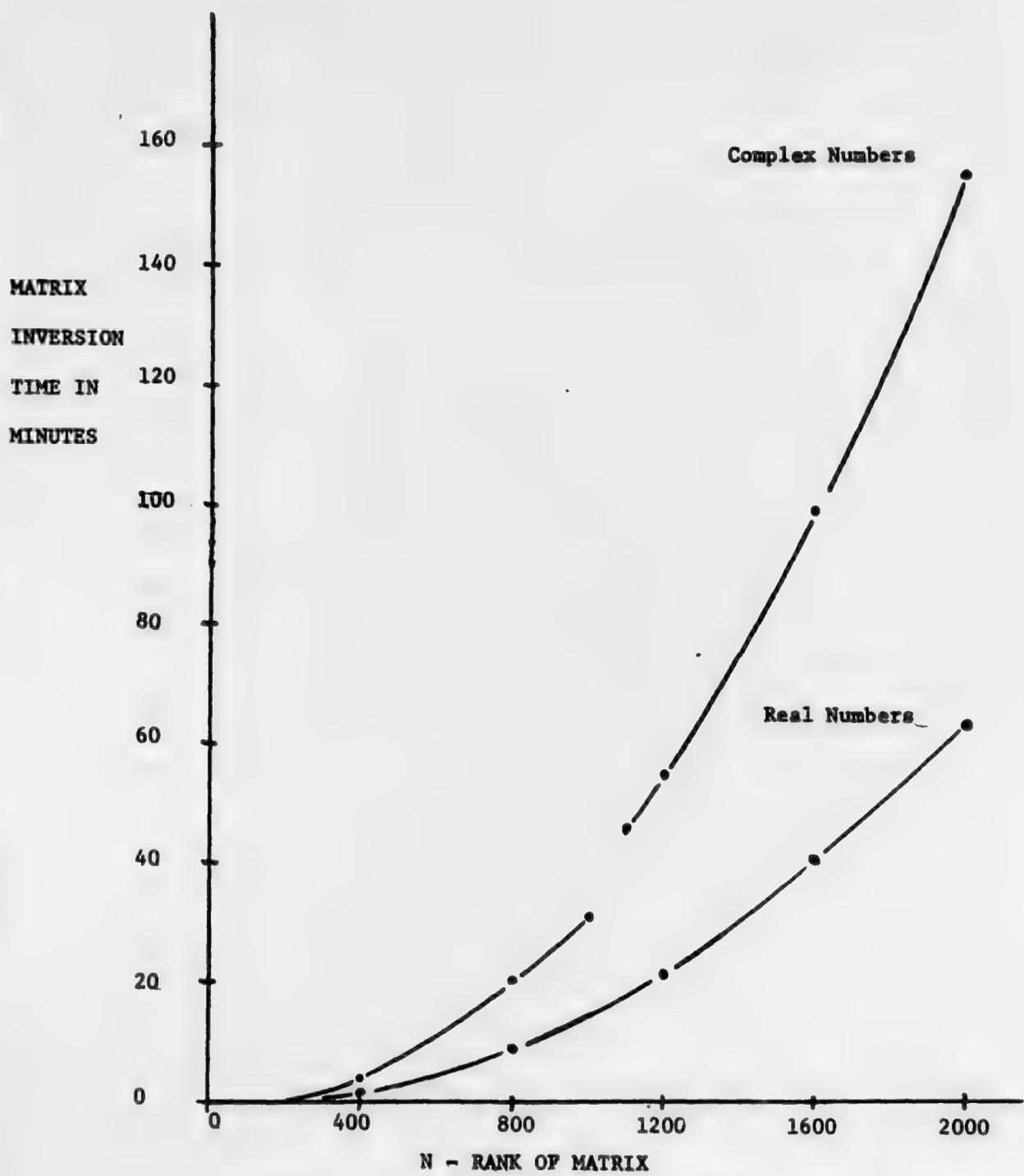


FIGURE 8. Matrix Inversion Time with 16 Arrays and Rank < 2000.

TABLE 5: Timing Data for Inverting a Matrix with Complex  
Numbers for  $100 < \text{Rank} < 2000$ . (16 Arrays)

<u>RANK</u>	<u>TIME</u>	
	<u>HRS.</u>	<u>MINS.</u>
100	0	0
200	0	1
300	0	2
400	0	4
500	0	7
600	0	10
700	0	15
800	0	20
900	0	25
1000	0	31
1100	0	46
1200	0	55
1300	1	5
1400	1	15
1500	1	26
1600	1	38
1700	1	51
1800	2	5
1900	2	19
2000	2	34

## CONCLUSIONS

In Table 6 are shown the inversion times, in minutes, for inverting both complex-numbered and real-numbered matrices of various sizes on the Goodyear/STARAN. Both a STARAN with 4 arrays (1024 words) and one with 16 arrays (4096 words) are considered. The inversion times for matrices with real numbers are taken from [1]. It is seen from the Table that complex-numbered matrices require about 2 1/2 to 4 times as long for inversion as do real-numbered matrices.

A number of assumptions have gone into the derivation of the above inversion times. These are:

1. The data corresponding to the matrix columns are available in auxiliary memory when and where desired,
2. Software floating point arithmetic is used,
3. The real and imaginary parts of a matrix element each require 32-bits, and
4. An AP word is 256 bits long.

Assumption 1 simply means that we have separated the data management problem from the realm of the inversion process. That is to say, the inversion times in Tables 3 through 6 do not include the clock time that would be spent in locating and accessing the desired data on auxiliary storage. A relaxation of this assumption may, in general, yield higher

TABLE 6: Comparative Inversion Times for Matrices with  
Complex Numbers and Matrices with Real Numbers.

4- Array		16-Array	
STARAN		STARAN	
	Complex	Real	Complex
<u>N</u>	<u>Numbers</u>	<u>Numbers</u>	<u>Numbers</u>
500	10	4	7
1000	76	19	31
1500	-	86	86
2000	-	152	154

	Minutes	Minutes
--	---------	---------

matrix inversion times than those computed; however, it may be possible to overlap the locating and accessing of data with the AP processing operations. The last observation remains largely untested.

On the other hand, as indicated in assumption 2, the computed inversion times reflect the software arithmetic that is being presently utilized on the STARAN. The Goodyear Aerospace Corporation has, for some time now, been considering hardware floating point arithmetic for the STARAN. Their estimated operation times for 32-bit-field-to-field add, subtract, multiply and divide are 40 usecs., 40 usecs., 42 usecs. and 110 usecs. respectively. The same operations, when performed common-register-to-field are estimated to require 29 usecs., 29 usecs., 29 usecs. and 97 usecs. respectively [1]. A comparison of these figures against those in Table 2 immediately indicates that the computed inversion times would be smaller if hardware, rather than software, arithmetic were available on the AP.

Assumptions 3 and 4 were made primarily to obtain the numerical timing data for the inversion process. The importance of these assumptions lies only in the fact that a minimum of seven working fields are required on the AP for efficiently performing complex-arithmetic operations in parallel (this should be clear from algorithms 1 and 2), as the assumptions allow; otherwise they are not crucial to our analysis at all.

It should also be mentioned here that it seems likely that techniques other than a full matrix inversion can be found that will reduce the computation time for solving systems of the form  $AX = B$  on an AP. This should especially be true if the A matrix displays some special characteristics, e.g., if it were sparse or diagonally dominant, etc. The computed inversion times consider only the Gauss-elimination method and do not take into account any special characteristics of the A matrix; they should consequently be taken as upper bounds on the inversion process carried out on an AP.

We are thus led to the conclusion that the results obtained thus far are promising and that it is reasonable to continue our investigations into the solution of these types of problems using STARAN like architectures.

REFERENCES

1. Berra, P.B. and Ashok K. Singhania, "Timing Figures for Inverting Large Matrices Using the STARAN Associative Processor.", RADC Report No. RADC-TR-75-73, March 1975, (ADA009643).

***MISSION***  
*of*  
***Rome Air Development Center***

RADC plans and conducts research, exploratory and advanced development programs in command, control, and communications (C<sup>3</sup>) activities, and in the C<sup>3</sup> areas of information sciences and intelligence. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.

